

Introduction

In our efforts to address the Big Data challenges of climate science, we are moving toward a notion of Climate Analytics-as-a-Service (CAaaS) and delivery of capabilities through the NASA Climate Data Services Application Programming Interface (CDS API). The CDS API provides a uniform semantic treatment of the combined functionalities of large-scale data management, data-proximal analytics, and related services. The CDS API combines concepts from the Open Archive Information Systems (OAIS) reference model, object-oriented programming APIs, and Web 2.0 resource-oriented APIs.

OAIS is sponsored by the Consultative Committee for Space Data Systems (CCSDS): <http://public.ccsds.org/publications/archive/650x0m2.pdf>

Download-Installation

CDS services are available by accessing the CDS API. The CDS API can be executed using either direct web service calls, a Python script of sequential commands, or a full featured Python application. The CDS client package contains everything needed to make CDS requests. Note, however, that while the client package abstracts web service calls for easy use with Python, it is not required for direct web services calls.

Source Code Download Page
The MERRA/AS client stack and API is released under the following disclaimer:

NASA's Goddard Space Flight Center
NASA/GSFC/CISTO
8800 Greenbelt Rd, Greenbelt, MD 20771

THIS SOFTWARE AND ITS DOCUMENTATION ARE CONSIDERED TO BE IN THE PUBLIC DOMAIN AND THUS ARE AVAILABLE FOR UNRESTRICTED PUBLIC USE. THEY ARE FURNISHED "AS IS." THE AUTHORS, THE UNITED STATES GOVERNMENT, ITS INSTRUMENTALITIES, OFFICERS, EMPLOYEES, AND AGENTS MAKE EXPRESS OR IMPLIED, AS TO THE USEFULNESS OF THE SOFTWARE AND DOCUMENTATION FOR ANY PURPOSE. THEY ASSUME NO RESPONSIBILITY (1) FOR USE OF THE SOFTWARE AND DOCUMENTATION; OR (2) TO PROVIDE TECHNICAL SUPPORT TO USERS.

Download Current Stable Release: [cds_client_0.9.0.tar](#)

Decompress the tar file (e.g., `tar -xvf cds_client_version.tar`) and see the enclosed 'usage.txt' for detailed usage.

Requirements

- Internet access
- Python 2.6+ (only required if accessing the CDS API via Python scripts or applications)
- Internet browser (only required if accessing the CDS API via direct web service URLs)

Climate Data Services - Application Programming Interface

The Climate Data Services (CDS) Application Programming Interface (API) defines the set of actions (e.g., methods) that can be requested. The CDS API is accessed according to the desired 'service' (i.e., MAS, PS). Each service exposes unique 'service requests' (e.g., `GetVariableByCollection_Operation_TimeRange_SpatialExtent_VerticalExtent`) that trigger independent logic within the service. Generally speaking, 'services' define categories of behavior while 'service requests' refer to specific actions within a specific service. Service requests tell the application what the requestor is trying to accomplish per service.

CDS-API-Method-Overview:

Each CDS API method, which corresponds to a specific web service URL context, is described below.

1. Order – Request data from a pre-determined service request (asynchronous).
2. Status – Track the progress of an order.
3. Download – Retrieve a Dissemination Information Package (DIP).
4. Ingest – Ingest a Submission Information Package (SIP).
5. Execute – Initiate a service-definable extension. Allows for parameterized growth without API change.
6. Query – Retrieve data using a pre-determined service request (synchronous).

Climate Data Services Library

The CDS Library is a Python class (i.e., `CDSApi`) that prepares web service calls while abstracting the details. This class, which can be imported into any Python application, includes basic utilities and extended utilities. Basic utilities call a single CDS API method (e.g., `order`) and can also be included in Python scripts and applications. Extended utilities may weave together multiple CDS API methods (e.g., `avg`) and are intended for use as convenience functions in Python scripts and applications. For sample invocations, see the MERRA/AS method overview.

All CDS requests ultimately pass through this single web service endpoint: <http://skyportal.sci.gsfc.nasa.gov/cds/mas/>

CDS Basic Utilities:

Basic utilities are designed for the CDS API Web Service calls, or Python scripts and applications.

Order

Request data using a pre-determined service request (asynchronous).

Typically initiates an asynchronous Map Reduce job that generates an aggregate NetCDF file according to the user-specified arguments.

Arguments:

service: The name of the target service. For example: MAS
request: The service request to be initiated (e.g., GetVariableByCollection_Operation_TimeRange_SpatialExtent_VerticalExtent
parameters: request-specific parameter list

Returns:

sessionId: session id. A unique UUID that is used to retrieve session status (e.g., 37734A627B5E7E74AAB8374586D3D018)
sessionStatus: One word status of the session (e.g., Received, Completed, Failed)
sessionStatusDetail: Description of the session (e.g., Order failed due to lack of privileges)

Order-Services:

Service	Description
MAS	MERRA/AS service, used to order and download subsetting MERRA/AS data

Status

Track the progress of an order.

Status is a method for session tracking. Once initiated, a session enters into different processing stages. The Status method offers the client a way of querying the current state of the session.

Arguments:

service: The name of the target service. For example: MAS
sessionId: Session ID

Returns:

sessionId: Session id
sessionStatus: One word status of the session (e.g., Processing, Completed, Failed)
sessionStatusDetail: Description of the session (e.g., MapReduce job in progress)

Status-Values:

Status	Description
Received	Order successfully received
Processing	Order in progress
Completed	Order/ingest was successfully completed
Failed	Order/ingest failed

Download

Retrieve a Dissemination Information Package (DIP).

The Download method provides a means for retrieving packages associated with a sessionId from a service provider. In a typical asynchronous transaction, such as those associated with Order, this method obtains results from a service request.

Arguments:

service: The name of the target service. For example: MAS
sessionId: Session ID for the order
DIP: Optional dissemination information package (name!type!content).
When this parameter is empty, all packages associated with the supplied sessionId are returned. When a package name is specified within the DIP parameter, the service provider returns only the associated package.

Returns:

DIP: Requested dissemination information package

Download-Services:

Service	Description
MAS	MERRA/AS service, used to order and download subsetting MERRA/AS data
PS	Persistence service, used to ingest, search for, and download a submission information package (SIP)

Ingest

Ingest a Submission Information Package (SIP).

Provide a SIP to be persisted by the service. Generally speaking, a Submission Information Package is simply an artifact to be stored, shared, and retrieved. With respect to CDS, a SIP will typically be a script or application to be shared, but any ingested artifact will be stored and available for download. A sample use case may involve ordering the average temperature for North America in 2013. The script that calculates this could be ingested and reused by other consumers that is interested in the same information. In this case, the consumer would simply download the script and run it locally. Additionally, accumulated scripts provide a head start for easy modifications to perform other operations (e.g., maximum, minimum).

Arguments:

service: The name of the target service. For example: MAS
operation: operation to perform (e.g., Put)
SIP: Submission information package (name!type!content)
sessionId: Optional session ID for the SIP if the operation is a result of a previous transaction (e.g., ingest parts of a single collection through multiple calls).
overwrite: If 'true' overwrite previous copy

Returns:

sessionId: Session id
sessionStatus: One word status of the session (e.g., Received, Completed, Failed)
sessionStatusDetail: Description of the session (e.g., Ingest failed due to insufficient HDFS storage space)

Ingest-Services:

Service	Description
PS	Persistence service, used to ingest, search for, and download a submission information package (SIP)

Execute

Initiate a service-definable extension to the classic CDS API. Allows for parameterized growth without API change.

The CDS API can be thought of as a plug-in architecture that allows services (e.g., MAS, PS) to be added and accessed in a common way. The Execute method is an optional strategy that allows service providers to add services without modifying the API. This extensibility is useful for CDS consumers that implement a single invocation of the Execute call on the web service tier, but vary parameters between calls. For applications that derive parameters from an external source (e.g., configuration file, database) this approach allows for parameter modifications without re-coding the application. It also allows service providers to expose functionality that is not OAIS-specific.

Arguments:

service: The name of the target service. For example: MAS
methodName: The name of the method to be processed. For example, GetFileNameByAttribute (methods supported by the Persistence
parameters: 1 or more parameters of the format (name!value)).

Returns:

sessionId: session id.
sessionStatus: One word status of the session (e.g., Received, Completed, Failed)
resultSet: execute results in XML format

Execute-Services:

Service	Description
PS	Persistence service, used to ingest, search for, and download a submission information package (SIP)

Query

Retrieve data using a pre-determined service request (synchronous).

The Query method executes a predefined service request (e.g., GetFileNameByAttribute) that returns data in an XML instance package. The CDS API suite supports a specific set of service requests that vary across services. One goal of the Query service is to provide dynamic information to the consumer that assists with CDS processing.

Arguments:

service: The name of the target service. For example: MAS
request: The name of the query to be processed. For example, GetFileNameByAttribute (queries supported by PS Service)
parameters: 1 or more parameters of the format (name!value).

Returns (resultSet or fault)

resultSet: query results in XML format OR
fault: Error code (e.g., E_InvalidParameter)

Query-Services:

Service	Description
PS	Persistence service, used to ingest, search for, and download a submission information package (SIP)